Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Computational and Applied Mathematics 170 (2004) 371–397

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.com/locate/cam

High-performance numerical algorithms and software for subspace-based linear multivariable system identification☆

Vasile Sima[a,*], Diana Maria Sima[b], Sabine Van Huffel[b]

[a] *National Institute for Research & Development in Informatics, Bd. Mareşal Averescu, No. 8-10, 011455 Bucharest 1, Romania*

[b] *Department of Electrical Engineering, ESAT-SCD(SISTA), Katholieke Universiteit Leuven, Kasteelpark Arenberg 10, B-3001 Leuven, Belgium*

Received 14 April 2003; received in revised form 18 December 2003

Abstract

Basic algorithmic and numerical issues involved in subspace-based linear multivariable discrete-time system identification are described. A new identification toolbox—SLIDENT—has been developed and incorporated in the freely available Subroutine Library in Control Theory (SLICOT). Reliability, efficiency, and ability to solve industrial identification problems received a special consideration. Two algorithmic subspace-based approaches (MOESP and N4SID) and their combination, and both standard and fast techniques for data compression are provided. Structure exploiting algorithms and dedicated linear algebra tools enhance the computational efficiency and reliability. Extensive comparisons with the available computational tools based on subspace techniques show the better efficiency of the SLIDENT toolbox, at comparable numerical accuracy, and its capabilities to solve identification problems with many thousands of samples and hundreds of parameters.

Keywords: Linear multivariable system; Numerical linear algebra; Parameter estimation; Subspace identification; Singular value decomposition

☆ This work was partly supported by the European Community BRITE-EURAM III *Thematic Networks Programme NICONET* (project BRRT-CT97-5040), by the Belgian Programme on Interuniversity Poles of Attraction (IUAP Phase V-22), initiated by the Belgian State, Prime Minister's Office—Federal Office for Scientific, Technical and Cultural Affairs, by the Concerted Research Action (GOA) projects of the Flemish Government MEFISTO-666 (Mathematical Engineering for Information and Communication Systems Technology), and by the FWO projects G.0078.01 and G.0270.02.

* Corresponding author.

E-mail address: vsima@u3.ici.ro (V. Sima).

0377-0427/$ - see front matter
doi:10.1016/j.cam.2003.12.046

1. Introduction

Consider a linear time-invariant (LTI) discrete-time state space model, described by

$$x_{k+1} = Ax_k + Bu_k + w_k,$$
$$y_k = Cx_k + Du_k + v_k, \tag{1}$$

where $x_k \in \mathbb{R}^n$ is the n-dimensional state vector at time k, $u_k \in \mathbb{R}^m$ is the input vector, $y_k \in \mathbb{R}^\ell$ is the output vector, A, B, C, and D are real matrices, and $\{w_k\}$, $\{v_k\}$ are zero mean, stationary ergodic state and output disturbance or noise sequences, uncorrelated with $\{u_k\}$ and with the initial state of (1), with covariances satisfying the relation

$$\mathscr{E}\left\{\begin{bmatrix} w_p \\ v_p \end{bmatrix} \begin{bmatrix} w_q^{\mathrm{T}} & v_q^{\mathrm{T}} \end{bmatrix}\right\} = \begin{bmatrix} Q & S \\ S^{\mathrm{T}} & R_v \end{bmatrix} \delta_{pq} \geqslant 0, \tag{2}$$

where $\mathscr{E}$ denotes the expected value operator and δ_{pq} is the Kronecker delta symbol. The matrix pair (A, C) is assumed observable, and $(A, (B \quad Q^{1/2}))$ controllable. A particular model, in *innovation form*, is

$$x_{k+1} = Ax_k + Bu_k + Ke_k,$$
$$y_k = Cx_k + Du_k + e_k, \tag{3}$$

where $\{e_k\}$ is a white noise sequence, and K is the *Kalman gain matrix*.

In *system identification* problems, the system order, n, and the quadruple of system matrices (A, B, C, D) have to be determined (up to a system similarity transformation) using the input and output data sequences, $\{u_k\}$ and $\{y_k\}$, $k = 1{:}t$ (i.e., for k taking integer values from 1 to t).[1] In addition, the Kalman gain matrix K in (3), as well as the state and output noise covariance matrices in (2) have often to be found.

An identified model can be used for various purposes, like analysis, simulation, fault detection, control, prediction, etc. For instance, given the initial estimate $\hat{x}_1$, and the trajectories $\{u_k\}$ and $\{y_k\}$, the predicted output can be computed recursively using the formulas

$$\hat{y}_k = \hat{C}\hat{x}_k + \hat{D}u_k,$$
$$\hat{x}_{k+1} = \hat{A}\hat{x}_k + \hat{B}u_k + \hat{K}(y_k - \hat{y}_k), \tag{4}$$

where the estimated quantities have been marked by hat signs (suppressed in the sequel for matrices, for convenience). If $\hat{K}$ is not available, the last term in (4) is omitted, but then the predicted output might not be very good, if there are significant disturbances.

Three basic subspace-based approaches have been proposed for solving system identification problems: Multivariable Output Error state SPace (MOESP) [28,30,31], Numerical algorithm for Subspace State Space System IDentification (N4SID) [18,25–27], and Canonical Variate Analysis (CVA) [12,19]. The main feature of the MOESP class of techniques is the determination of an extended

[1] It is assumed that t is the number of data measurements in the *estimation set*, used to compute the estimates; additional data, in the *validation set*, could be used to check the results. In theory, $t \to \infty$, and the results hold asymptotically. The practical implication is the need to use large enough data sets, so that the asymptotic results hold approximately.

observability matrix of the deterministic part of model (1). The main feature of the N4SID class of techniques is the determination of the estimated state sequence of the LTI system via the intersection of, or projection on the row spaces of the Hankel-like matrices constructed from "past" and "future" input–output (I/O) data. The CVA techniques also use the state sequence, in a statistical framework. A recent survey of subspace methods is [5]. Several variants of these basic subspace approaches have been recently proposed, for instance, in [16,19]. However, the paper focuses on MOESP and N4SID, because software based on these techniques are widely available and often used, thus enabling to make performance comparisons. In addition, the underlying theory is well developed and the related numerical issues are well understood. Moreover, it can be shown that many subspace algorithms, including those based on MOESP, N4SID, and CVA approaches, use the same subspace, but different weightings, to find the order and the extended observability matrix. See [5] and the references therein.

Subspace-based system identification approaches are attractive for several reasons: state-space models are directly estimated; no parameterizations are needed; robust linear algebra tools like QR decomposition and singular value decomposition (SVD) are used; only one parameter, s, has to be chosen. The delivered results could be used, when needed, to initialize optimization-based approaches, like prediction error methods (PEM) [14].

Subspace identification algorithms start by building a *block-Hankel-block matrix* H, from concatenated block-Hankel matrices, using (part of) the available I/O data. For MOESP with past inputs and outputs, H is given by

$$H = [U_{p+1,\,p+s,N+p}^{\mathrm{T}} \;\; U_{1,\,p,N}^{\mathrm{T}} \;\; Y_{1,q+s,N}^{\mathrm{T}}] \in \mathbb{R}^{N\times(mp+\ell q+(m+\ell)s)}, \tag{5}$$

where $N = t - \min(p,q) - s + 1$ (usually $N \gg (mp + \ell q + (m+\ell)s)$),

$$U_{a,b,c} = \begin{bmatrix} u_a & u_{a+1} & u_{a+2} & \cdots & u_c \\ u_{a+1} & u_{a+2} & u_{a+3} & \cdots & u_{c+1} \\ u_{a+2} & u_{a+3} & u_{a+4} & \cdots & u_{c+2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ u_b & u_{b+1} & u_{b+2} & \cdots & u_{c+b-a} \end{bmatrix} \tag{6}$$

and similarly for $Y_{a,b,c}$. For N4SID, the first two block columns of H are interchanged, so they can be rewritten as $U_{1,\,p+s,N}^{\mathrm{T}}$. The submatrices $U_{1,\,p,N}$ and $Y_{1,q,N}$ are called the "past" input and output parts, respectively, while the submatrices $U_{p+1,\,p+s,N+p}$ and $Y_{q+1,q+s,N+q}$ are called the "future" parts. The latest version of the MATLAB[2] identification toolbox [15] uses different "prediction horizons" s, p, and q, whose default values are chosen using the Akaike information criterion (AIC). But standard MOESP and N4SID algorithms, discussed in the sequel, use $p = q = s$, where s denotes the "number of block rows", and s should satisfy $s \geqslant n$.[3] Therefore, for convenience, it will be

[2] MATLAB is a trademark of The MathWorks, Inc.

[3] The MOESP theory [30,31] assumes $s > n$. In practice, s is often chosen larger than the system order n, e.g., $s = 2n$. A large value of s usually produces more accurate results. It is also possible to choose the prediction horizons for past data smaller than n, for instance, $p = q > n/(m+\ell)$ [33]. The minimum values accepted by the `n4sid` function in [15] are $s = \lceil n/\ell \rceil + 1$ and $p = q = \lceil (n - \ell + 1)/(m+\ell) \rceil$, where $\lceil a \rceil$ denotes the nearest integer larger than or equal to a. Worse results are normally obtained with these minimum values than with larger ones.

assumed that

$$H = \begin{cases} [U_{s+1,2s,N+s}^{\mathrm{T}} \;\; U_{1,s,N}^{\mathrm{T}} \;\; Y_{1,2s,N}^{\mathrm{T}}] & \text{for MOESP,} \\ [U_{1,2s,N}^{\mathrm{T}} \;\; Y_{1,2s,N}^{\mathrm{T}}] & \text{for N4SID,} \end{cases} \qquad H \in \mathbb{R}^{N \times 2(m+\ell)s}, \tag{7}$$

where $N = t - 2s + 1$, and $U_{\cdot,\cdot,\cdot}$ and $Y_{\cdot,\cdot,\cdot}$ are defined by formulas like (6). The CVA algorithms [12,19] essentially make use of the covariance matrix $H^{\mathrm{T}}H$.

Given H (or H^{T}), the algorithms find an upper (or lower) triangular factor R (or L) from a QR (or LQ) factorization of H (or H^{T}).[4] Parts of R are further used to estimate n and system and covariance matrices. MATLAB and Fortran codes implementing the MOESP and N4SID approaches have been developed, e.g., [10,15,21,24,25,29]. The Fortran codes include an option to process the I/O data either in a single batch, or in multiple data batches, which is important from a practical point of view. These two strategies are referred to as *nonsequential*, and *sequential data processing*, respectively.

Since the number of measurements, t, can be very large, matrix H is often huge and the identification problem has high computational complexity. Therefore, it is of paramount importance to provide fast and reliable algorithms and associated software for its solution. This was the main objective of developing the new subspace-based linear system identification toolbox [22]—SLIDENT—incorporated in the **S**ubroutine **L**ibrary **I**n **CO**ntrol **T**heory (SLICOT)[5] [3]. SLICOT is based on the state-of-the-art linear algebra package LAPACK [2], and on the Basic Linear Algebra Subprograms (BLAS) collections [13,7,6], and, therefore, it can benefit of the advanced features of modern computer architectures. Moreover, the new routines include algorithmic improvements and refinements over the previous implementations. For instance, the particular structure of the block-Hankel-block matrix H can be exploited when computing the R factor of its QR factorization, and this could speed-up the calculations by more than an order of magnitude. Last but not least, MATLAB and Scilab [9] interfaces, consisting in MEX-files and M-files, are provided in SLIDENT, in order to increase the user-friendliness.

The aim of this paper is to describe those theoretical and mainly algorithmical advances which proved to be most useful for developing the associated high-performance software for solving system identification problems. The availability of powerful problem solving tools in this area is very important in practice, since modern control techniques are heavily dependent on suitable dynamical models. The paper is organized as follows. After a short overview of the mathematical foundations (Section 2), several numerical techniques for data compression are presented in Section 3.1: sequential QR factorization of the matrix H, Cholesky factorization of the associated inter-correlation matrix $H^{\mathrm{T}}H$, thereby exploiting the block-Hankel structure, and a fast QR factorization technique, based on the displacement structure. Finding the system order and the estimation of system matrices are also discussed in Sections 3.2 and 3.3, respectively. Most of the results are described or derived in a new, simpler way, based on numerical linear algebra arguments. Numerical results obtained using the SLIDENT toolbox components on a large collection of relevant data sets are then summarized in Section 4, illustrating the high efficiency of the developed tools. Finally, the main abilities

[4] Theoretical work uses H^{T} and the lower triangular factor L, but all known implementations use H and R.

[5] SLICOT Library is freely available for noncommercial applications from the NICONET Web site address: `http://www.win.tue.nl/wgs/niconet.html`.

of the SLIDENT toolbox are briefly described in Appendix, indicating the function of the essential MATLAB/Scilab interfaces.

2. Mathematical foundations

Essential facts about row space projections, which are used in subspace-based identification techniques, are first reviewed. Let $E \in \mathbb{R}^{p\times N}$, $F \in \mathbb{R}^{q\times N}$, $G \in \mathbb{R}^{r\times N}$, and let

$$E/F = EF^{\dagger}F =: E\mathscr{P}_F \tag{8}$$

denote the projection of the row space of E into the row space of F (with $\cdot^{\dagger}$ the Moore–Penrose pseudo-inverse), and let $F^{\perp}$ denote the orthogonal complement of the row space of F (or a basis for it). Clearly,

$$E/F^{\perp} = E(I - \mathscr{P}_F) = E - E/F, \tag{9}$$

$$E/F = \begin{cases} EF^{\mathrm{T}}(FF^{\mathrm{T}})^{-1}F & \text{if } F \text{ has full row rank,} \\ EF^{\mathrm{T}}(FF^{\mathrm{T}})^{\dagger}F & \text{otherwise.} \end{cases} \tag{10}$$

Define also the *oblique projection* of the row space of E into the row space of F along the row space of G,

$$E/_G F = (E/G^{\perp})(F/G^{\perp})^{\dagger}F. \tag{11}$$

The following lemma collects some results about projections, which will be used later.

Lemma 1. *Let $E \in \mathbb{R}^{p\times N}$, $F \in \mathbb{R}^{q\times N}$, and E/F the projection involving row spaces.*

(i) *If $E = \tilde{E}Q$ and $F = \tilde{F}Q$, where Q is (square) orthogonal or has orthonormal rows, i.e., $QQ^{\mathrm{T}} = I$, then $E/F = (\tilde{E}/\tilde{F})Q$, i.e., the projection is postmultiplied by Q when the matrices are postmultiplied by such a matrix Q. A similar result holds for an oblique projection.*

(ii) *If $E = [E_1\ 0]$ and $F = [F_1\ 0]$, where E_1 and F_1 have the same number of columns, then $E/F = [E_1/F_1\ 0]$.*

(iii) *Let $F = U\Sigma V^{\mathrm{T}} = U_1\Sigma_1 V_1^{\mathrm{T}}$ be the SVD of F, where $U = [U_1\ U_2]$ and $V = [V_1\ V_2]$ are orthogonal, $\Sigma_1 \in \mathbb{R}^{k\times k}$ is diagonal nonsingular, and $U_1 \in \mathbb{R}^{q\times k}$, $V_1 \in \mathbb{R}^{N\times k}$, with $k = \operatorname{rank}(F) \leqslant \min\{q, N\}$. Then,*

$$E/F = EV_1V_1^{\mathrm{T}}, \quad \rho := E/F^{\perp} = EV_2V_2^{\mathrm{T}}, \quad \rho(E/F)^{\mathrm{T}} = 0. \tag{12}$$

Moreover, if $r := E^{\mathrm{T}} - F^{\mathrm{T}}\underline{X}$ is the residual *corresponding to the minimum norm solutions $\underline{x}_i$, $\underline{X} = [\underline{x}_1\ \cdots\ \underline{x}_p]$, of the least-squares problems $\min_{x_i}\|F^{\mathrm{T}}x_i - e_{i,1:N}\|_2$, $i = 1{:}p$, then $\rho = r^{\mathrm{T}}$.*

Proof. The results in (i) and (ii) follow easily from the definition of E/F. For instance,

$$E/F = EF^{\mathrm{T}}(FF^{\mathrm{T}})^{\dagger}F = \tilde{E}\tilde{F}^{\mathrm{T}}(\tilde{F}\tilde{F}^{\mathrm{T}})^{\dagger}\tilde{F}Q = (\tilde{E}/\tilde{F})Q.$$

Relations (12) follow from the definition of E/F and the SVD of F, which gives $F^\dagger = V_1\Sigma_1^{-1}U_1^{\mathrm{T}}$. For the last result, since $\underline{X} = (F^{\mathrm{T}})^\dagger E^{\mathrm{T}} = U_1\Sigma_1^{-1}V_1^{\mathrm{T}}E^{\mathrm{T}}$, then,

$$r = E^{\mathrm{T}} - V_1\Sigma_1U_1^{\mathrm{T}}U_1\Sigma_1^{-1}V_1^{\mathrm{T}}E^{\mathrm{T}} = (I - V_1V_1^{\mathrm{T}})E^{\mathrm{T}} = V_2V_2^{\mathrm{T}}E^{\mathrm{T}} = \rho^{\mathrm{T}}. \qquad \square$$

Define now the past and future parts of the input and output data for system (1)

$$U_{\mathrm{p}} = U_{1,s,N}, \quad Y_{\mathrm{p}} = Y_{1,s,N}, \quad U_{\mathrm{f}} = U_{s+1,2s,N+s}, \quad Y_{\mathrm{f}} = Y_{s+1,2s,N+s} \tag{13}$$

and similarly define the block-Hankel matrices for w_k and v_k as M_* and N_*, respectively, with $* \in \{\mathrm{p},\mathrm{f}\}$. From (1), with $k = 1:N$, it follows easily that

$$Y_* = \Gamma_s X_* + H_s U_* + H_s^w M_* + N_*, \quad * \in \{\mathrm{p},\mathrm{f}\}, \tag{14}$$

$$\Gamma_s := [C^{\mathrm{T}}\ (CA)^{\mathrm{T}}\ (CA^2)^{\mathrm{T}}\ \cdots\ (CA^{s-1})^{\mathrm{T}}]^{\mathrm{T}}, \tag{15}$$

$$X_{\mathrm{p}} := [x_1\ x_2\ x_3\ \cdots\ x_N], \quad X_{\mathrm{f}} := [x_{s+1}\ x_{s+2}\ x_{s+3}\ \cdots\ x_{s+N}],$$

where H_s and H_s^w are lower block triangular Toeplitz matrices of *Markov parameters*

$$H_s = \begin{bmatrix} D & 0 & 0 & \cdots & 0 \\ CB & D & 0 & \cdots & 0 \\ CAB & CB & D & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ CA^{s-2}B & CA^{s-3}B & \cdots & \cdots & D \end{bmatrix}, \quad H_s^w = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ C & 0 & 0 & \cdots & 0 \\ CA & C & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ CA^{s-2} & CA^{s-3} & \cdots & \cdots & 0 \end{bmatrix}. \tag{16}$$

Taking $t \to \infty$ in (14), and using also the assumed properties of $\{w_k\}$ and $\{v_k\}$, gives

$$Y_{\mathrm{f}}/U_{\mathrm{f}}^{\perp} = \Gamma_s X_{\mathrm{f}}/U_{\mathrm{f}}^{\perp} + H_s^w M_{\mathrm{f}} + N_{\mathrm{f}},$$

since $U_{\mathrm{f}}/U_{\mathrm{f}}^{\perp} = 0$ and the noise is uncorrelated with the input sequence. Pre- and post-multiplying with suitable weighting matrices W_1 and W_2, chosen so that

- $\mathrm{rank}(W_1\Gamma_s) = \mathrm{rank}(\Gamma_s)$,
- $\mathrm{rank}(X_{\mathrm{f}}) = \mathrm{rank}(X_{\mathrm{f}}/U_{\mathrm{f}}^{\perp}W_2)$,
- $M_{\mathrm{f}}W_2 = 0,\ N_{\mathrm{f}}W_2 = 0$,

it follows that

$$\mathcal{O}_s := W_1Y_{\mathrm{f}}/U_{\mathrm{f}}^{\perp}W_2 = W_1\Gamma_sX_{\mathrm{f}}/U_{\mathrm{f}}^{\perp}W_2. \tag{17}$$

From the SVD of $\mathcal{O}_s$,

$$\mathcal{O}_s = U\Sigma V^{\mathrm{T}} = [U_1\ U_2]\begin{bmatrix} S_1 & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} V_1^{\mathrm{T}} \\ V_2^{\mathrm{T}} \end{bmatrix}, \tag{18}$$

we have $n = \mathrm{rank}(\mathcal{O}_s)$, and we may take (modulo a system similarity transformation)

$$W_1\Gamma_s = U_1S_1^{1/2}, \quad \tilde{X}_s := X_{\mathrm{f}}/U_{\mathrm{f}}^{\perp}W_2 = S_1^{1/2}V_1^{\mathrm{T}}. \tag{19}$$

Standard choices are

$$W_1 = \begin{cases} I_{\ell s} & \text{for MOESP and N4SID,} \\ [(Y_f/U_f^{\perp})(Y_f/U_f^{\perp})^T]^{-1/2} & \text{for CVA,} \end{cases} \tag{20}$$

$$W_2 = \begin{cases} (W_p/U_f^{\perp})^{\dagger}(W_p/U_f^{\perp}) & \text{for MOESP and CVA,} \\ (W_p/U_f^{\perp})^{\dagger}W_p & \text{for N4SID,} \end{cases} \tag{21}$$

where $W_p := [U_p^T \; Y_p^T]^T$. Based on the presentation above, the following results can be proven for the N4SID approach [5,26], but similar results hold for the MOESP approach [28,30], or for the CVA approach [19]. See also [33], for a general statistical analysis of identification methods based on instrumental variables and "subspace fitting" and the asymptotic properties of subspace estimates.

Theorem 2 (Main subspace identification theorem). *Assuming that*:

1. $\{u_k\}$ *is uncorrelated with* $\{w_k\}$ *and* $\{v_k\}$;
2. $\{u_k\}$ *is* persistently exciting of order $2s$, *i.e.*, $\operatorname{rank}(U_{1,2s,N}U_{1,2s,N}^T) = 2ms$;
3. $t \to \infty$;
4. $\{w_k\}$ *and* $\{v_k\}$ *are not identically* 0;
5. $W_2 = (W_p/U_f^{\perp})^{\dagger}W_p$ (*i.e.*, *the N4SID scheme*);

then

1. *The system order equals the number of nonzero singular values of* $\mathcal{O}_s$.
2. *The weighted projection* $\mathcal{O}_s$ *can be written as the* oblique projection of future outputs into the past inputs and outputs along the future inputs, $\mathcal{O}_s = W_1 Y_f/_{U_f} W_p$, *and it can be factored as* $\mathcal{O}_s = W_1\Gamma_s\tilde{X}_s$, *where* Γ_s *is the* extended observability matrix *and* $\tilde{X}_s$ *is a* Kalman filter estimated state sequence of X_f.
3. Γ_s *and* $\tilde{X}_s$ *can be recovered from* $\Gamma_s = W_1^{-1}U_1S_1^{1/2}$ *and* $\tilde{X}_s = S_1^{1/2}V_1^T$.

In [26], it is shown that the columns of $\tilde{X}_s$ can be interpreted as the states of a bank of N nonsteady state Kalman filters applied in parallel to the data, with the same initial error covariance matrix and appropriate initial conditions. This interpretation motivates the name for the sequence $\tilde{X}_s$. Note that $\mathcal{O}_s$ in Theorem 2 is equivalent to the definition in (17), due to (11), (20) and (21). It should also be stressed that all singular values of $\mathcal{O}_s$ in (18) are normally nonzero, and "far" from zero for stochastic systems. It is usually difficult to decide the rank of $\mathcal{O}_s$, that is, the system order. The calculation of $\mathcal{O}_s$ is described in Section 3.2.

The matrices C and A can be estimated using the *shift invariance* property of Γ_s, i.e.,

$$\Gamma_s = \begin{bmatrix} C \\ \Gamma_{s-1}A \end{bmatrix}, \quad \text{hence } C = \Gamma_s(1{:}\ell, :), \quad A = \underline{\Gamma_s}^{\dagger}\overline{\Gamma_s}, \tag{22}$$

where $\overline{\Gamma_s}$ and $\underline{\Gamma_s} = \Gamma_{s-1}$ denote Γ_s without the first and last ℓ rows, respectively.

In principle, all system matrices could be found from the least-squares problem

$$\begin{bmatrix} \tilde{X}_{s+1} \\ Y_{s+1,s+1,N+s} \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} \tilde{X}_s \\ U_{s+1,s+1,N+s} \end{bmatrix} + \begin{bmatrix} \rho_w \\ \rho_v \end{bmatrix} \tag{23}$$

(generally giving biased estimates [26]), and the covariance matrices approximated by

$$\begin{bmatrix} Q & S \\ S^{\mathrm{T}} & R_v \end{bmatrix} \approx \frac{1}{N} \begin{bmatrix} \rho_w \\ \rho_v \end{bmatrix} [\rho_w^{\mathrm{T}} \ \rho_v^{\mathrm{T}}]. \tag{24}$$

The practical calculations are more involved, to enhance the efficiency and reliability.

3. Algorithms outline

The following procedure can be used for performing subspace-based system identification calculations.

Basic subspace identification procedure

(1) *Input–output data processing*: Construct (explicitly or implicitly) H in (7), and perform a "data compression" by computing the upper triangular factor R of a QR factorization, $H = \mathcal{Q}[R^{\mathrm{T}}\ 0]^{\mathrm{T}} = QR$; the matrix $\mathcal{Q} = [Q\ Q_2]$, $Q \in \mathbb{R}^{N \times 2(m+\ell)s}$, is not needed.

(2) *Finding system order*: Compute the SVD of a matrix R_o, built from R, where

$$R_o := \begin{cases} R_{ms+1:(2m+\ell)s,(2m+\ell)s+1:2(m+\ell)s}^{\mathrm{T}} & \text{for MOESP,} \\ \mathcal{O}_s & \text{for N4SID.} \end{cases} \tag{25}$$

The number of "nonzero" singular values gives the order n of the system.

(3) *Finding system matrices and Kalman gain*: Find system matrices from the left singular vectors U, and other submatrices of R.

Find covariance matrices using the residuals of a least-squares problem.

Find the Kalman gain by solving a discrete-time algebraic matrix Riccati equation for the Kalman filtering problem corresponding to (1) and (2).

It should be emphasized that the order selection strategy in Step (2) only holds for the ideal case of identification of a LTI finite-dimensional system, and for $t \to \infty$. The above procedure will be detailed in the next subsections, also indicating how the problem structure can be exploited.

3.1. Input–output data processing

The data used for subspace system identification is basically given as two matrices, $U \in \mathbb{R}^{t \times m}$ and $Y \in \mathbb{R}^{t \times \ell}$, for inputs and outputs, respectively. Since sometimes the data come in batches, provisions have been taken to enable sequential processing of the I/O measurements. For standard nonsequential data processing, the $N \times 2(m+\ell)s$ block-Hankel-block matrix H is constructed, and a standard QR factorization [8], $H = QR$, is used for data compression. The triangular factor R is a $2(m+\ell)s \times 2(m+\ell)s$ matrix. For sequential data processing, the QR factorization is done sequentially, by updating the upper triangular factor R.

Besides these QR algorithms, SLIDENT includes fast Cholesky and fast QR factorization algorithms, which exploit the special structure of the matrix H. The implemented algorithms differ from those discussed, e.g., in [4,11,17].

3.1.1. Sequential QR factorization

Consider first that there are only two data batches, (U_1, Y_1) and (U_2, Y_2), and let H_1 and H_2 be the corresponding block-Hankel-block matrices, with N_1 and N_2 rows. The QR-based algorithms first compute the QR factorization of H_1, $H_1 = Q_1 R_1$. Then, the upper triangular factor is updated using a specialized QR factorization for the problem $[R_1^{\mathrm{T}}\ H_2^{\mathrm{T}}]^{\mathrm{T}} = QR$, where the upper triangular structure of R_1 is exploited. One uses $2(m+\ell)s$ Householder transformations [8] of the same order, $1+N_2$; each Householder transformation i annihilates all the elements of the ith column of H_2, modifying the (i,i) element of R_1. Clearly, QR is a QR factorization of $H = [H_1^{\mathrm{T}}\ H_2^{\mathrm{T}}]^{\mathrm{T}}$. If there are additional data batches, the same procedure is applied repeatedly.

3.1.2. Cholesky factorization and block-Hankel structure exploitation

In order to use the Cholesky factorization algorithm [8] for data compression, one should first build the inter-correlation matrix $W = H^{\mathrm{T}}H$, and then factor W, assuming it is positive definite (which is usually the case in practice, due to various noise components). For nonsequential processing using the N4SID approach, the block-Hankel matrices corresponding to the inputs and outputs are $H_u = U_{1,2s,N}^{\mathrm{T}}$ and $H_y = Y_{1,2s,N}^{\mathrm{T}}$, and $H = [H_u\ H_y]$. The case of the MOESP approach is summarized near the end of this subsection (see Lemma 4). The definitions above can be extended for multiple batches. Actually, assuming that the latest batch to be processed is defined by H, then $W = \tilde{W} + H^{\mathrm{T}}H$, where $\tilde{W}$ corresponds to the already processed data batches. Clearly, $\tilde{W} = 0$ if the first (or single) data batch is processed.

The following lemma shows how the symmetric block matrix W can be computed exploiting the block-Hankel structure.

Lemma 3 (Structure-exploiting computation of W for N4SID approach). *Define* $W_{uu} = \tilde{W}_{uu} + H_u^{\mathrm{T}}H_u$, $W_{uy} = \tilde{W}_{uy} + H_u^{\mathrm{T}}H_y$, *and* $W_{yy} = \tilde{W}_{yy} + H_y^{\mathrm{T}}H_y$, *where each block consists of* $2s \times 2s$ *submatrices of sizes* $m \times m$, $m \times \ell$, *and* $\ell \times \ell$, *respectively. Denoting* $W_{uu}^{i,j}$ *the submatrix* (i,j) *of* W_{uu} $(j = 1{:}2s, i = 1{:}2s)$, *and similarly for* W_{uy} *and* W_{yy}, *then*

$$W_{uu}^{i,j} = \tilde{W}_{uu}^{i,j} + u_i u_j^{\mathrm{T}} + u_{i+1}u_{j+1}^{\mathrm{T}} + \cdots + u_{i+N-1}u_{j+N-1}^{\mathrm{T}}, \quad i = 1, \forall j, \tag{26}$$

$$W_{uu}^{i+1,j+1} = \tilde{W}_{uu}^{i+1,j+1} - \tilde{W}_{uu}^{i,j} + W_{uu}^{i,j} + u_{i+N}u_{j+N}^{\mathrm{T}} - u_i u_j^{\mathrm{T}}, \quad j = 1{:}2s-1,\ i = 1:j, \tag{27}$$

$$W_{uy}^{i,j} = \tilde{W}_{uy}^{i,j} + u_i y_j^{\mathrm{T}} + u_{i+1}y_{j+1}^{\mathrm{T}} + \cdots + u_{i+N-1}y_{j+N-1}^{\mathrm{T}}, \quad i = 1, \forall j,\ \text{or}\ j = 1, \forall i, \tag{28}$$

$$W_{uy}^{i+1,j+1} = \tilde{W}_{uy}^{i+1,j+1} - \tilde{W}_{uy}^{i,j} + W_{uy}^{i,j} + u_{i+N}y_{j+N}^{\mathrm{T}} - u_i y_j^{\mathrm{T}}, \quad i,j = 1{:}2s-1 \tag{29}$$

and $W_{yy}^{i,j}$ *is given by formulas similar to* (26) *and* (27), *with u replaced by y.*

Proof. Formulas (26) and (28) follow from the definition of W_{uu} and W_{uy}, respectively, and are valid for all $i,j = 1{:}2s$. Formulas (27) and (29) follow from (26) and (28), respectively, using the block-Hankel structure. (This is easily seen, for instance, when forming $W_{uu}^{i+1,j+1} - W_{uu}^{i,j}$.) □

For efficiency, in practical Fortran calculations, the upper triangle of W_{uu} is computed column-wise, i.e., the submatrices are processed in the order $W_{uu}^{1,1}$, $W_{uu}^{1,2}$, $W_{uu}^{2,2},\ldots,W_{uu}^{2s,2s}$, and similarly for the block W_{yy}. The submatrices $W_{uy}^{i,j}$ are also computed column-wise, before computing W_{yy}. Clearly, all but $8s$ submatrices are computed using updating formulas, such as (27) and (29), which are much cheaper (for large N) than (26) and (28).

Lemma 4 (Structure-exploiting computation of W for MOESP approach). *Let W be the matrix computed in Lemma* 3, *and let W_M denote the corresponding matrix for the MOESP approach. Partition W*

$$W = \begin{bmatrix} W_{11} & W_{12} & W_{13} \\ W_{12}^{\mathrm{T}} & W_{22} & W_{23} \\ W_{13}^{\mathrm{T}} & W_{23}^{\mathrm{T}} & W_{yy} \end{bmatrix}, \quad \text{where } W_{11}, W_{12}, W_{22} \in \mathbb{R}^{ms\times ms},\ W_{13}, W_{23} \in \mathbb{R}^{ms\times 2\ell s}$$

and partition W_M similarly. Then,

$$W_M = \begin{bmatrix} W_{22} & W_{12}^{\mathrm{T}} & W_{23} \\ W_{12} & W_{11} & W_{13} \\ W_{23}^{\mathrm{T}} & W_{13}^{\mathrm{T}} & W_{yy} \end{bmatrix}. \tag{30}$$

Proof. The result follows from the relation between the block-Hankel-block matrices for MOESP and N4SID approaches, namely

$$H_{\mathrm{MOESP}} = H_{\mathrm{N4SID}} \begin{bmatrix} 0 & I_{ms} & 0 \\ I_{ms} & 0 & 0 \\ 0 & 0 & I_{2\ell s} \end{bmatrix}. \tag{31}$$

Finally, the Cholesky factorization algorithm is applied to the computed matrix W. In the rare case when this fast algorithm fails, the QR factorization is automatically used, for nonsequential processing. This is not possible for sequential processing because not all the data are then available, but the calculations could be restarted using the QR algorithm.

3.1.3. Fast QR algorithm

First, consider the N4SID approach. The matrix W is symmetric positive semi-definite. Define the *shift matrix* $Z = \mathrm{diag}(Z_u, Z_y)$, where Z_u (Z_y) is a $2s \times 2s$ block matrix with $m \times m\,(\ell \times \ell)$ blocks, all zero except for identity blocks on the superdiagonal,

$$Z_u = \begin{bmatrix} 0_m & I_m & & \\ & 0_m & \ddots & \\ & & \ddots & I_m \\ & & & 0_m \end{bmatrix}, \quad Z_y = \begin{bmatrix} 0_\ell & I_\ell & & \\ & 0_\ell & \ddots & \\ & & \ddots & I_\ell \\ & & & 0_\ell \end{bmatrix}.$$

The next lemma shows that the symmetric matrix $\nabla W = W - Z^{\mathrm{T}} W Z$, called the *displacement* of W [11], has a low rank factorization.

Lemma 5. *The displacement of W can be written as*

$$\nabla W = G^{\mathrm{T}} \Sigma G, \quad \Sigma = \mathrm{diag}(I_p, -I_q), \tag{32}$$

where $p = q = m + \ell + 1$, hence ∇W has the rank $2(m + \ell + 1)$ at most. G is called the generator *of W.*

Proof. The proof assumes that $\tilde{W} = 0$, since the extension to the general case is trivial. Partition $U_{1,2s,N} =: H_u^{\mathrm{T}}$ in (6) as follows:

$$U_{1,2s,N} = \begin{bmatrix} u_1 & u_2 & \cdots & u_N \\ u_2 & u_3 & \cdots & u_{N+1} \\ \vdots & \vdots & \vdots & \vdots \\ u_{2s} & u_{2s+1} & \cdots & u_{N+2s-1} \end{bmatrix} = \begin{bmatrix} \hat{u}_1 & \hat{u}_2 \\ \hat{u}_3 & \hat{u}_4 \end{bmatrix} = \begin{bmatrix} \tilde{u}_1 & \tilde{u}_2 \\ \tilde{u}_3 & \tilde{u}_4 \end{bmatrix},$$

where $\hat{u}_1 \in \mathbb{R}^{m \times (N-1)}$, $\hat{u}_2 \in \mathbb{R}^m$, $\hat{u}_3 \in \mathbb{R}^{(2s-1)m \times (N-1)}$, $\hat{u}_4 \in \mathbb{R}^{(2s-1)m}$, $\tilde{u}_1 \in \mathbb{R}^{(2s-1)m}$, $\tilde{u}_2 \in \mathbb{R}^{(2s-1)m \times (N-1)}$, $\tilde{u}_3 \in \mathbb{R}^m$, $\tilde{u}_4 \in \mathbb{R}^{m \times (N-1)}$. Defining ∇W_u as the upper left $2ms \times 2ms$ submatrix of ∇W, corresponding to $W_{uu} = H_u^{\mathrm{T}} H_u$ and Z_u, it follows that

$$\begin{aligned} \nabla W_u &= \begin{bmatrix} \hat{u}_1 \hat{u}_1^{\mathrm{T}} + \hat{u}_2 \hat{u}_2^{\mathrm{T}} & \hat{u}_1 \hat{u}_3^{\mathrm{T}} + \hat{u}_2 \hat{u}_4^{\mathrm{T}} \\ \hat{u}_3 \hat{u}_1^{\mathrm{T}} + \hat{u}_4 \hat{u}_2^{\mathrm{T}} & \hat{u}_3 \hat{u}_3^{\mathrm{T}} + \hat{u}_4 \hat{u}_4^{\mathrm{T}} \end{bmatrix} - \begin{bmatrix} 0 & 0 \\ 0 & \tilde{u}_1 \tilde{u}_1^{\mathrm{T}} + \tilde{u}_2 \tilde{u}_2^{\mathrm{T}} \end{bmatrix} \\ &= \begin{bmatrix} \hat{u}_1 \hat{u}_1^{\mathrm{T}} + \hat{u}_2 \hat{u}_2^{\mathrm{T}} & \hat{u}_1 \hat{u}_3^{\mathrm{T}} + \hat{u}_2 \hat{u}_4^{\mathrm{T}} \\ \hat{u}_3 \hat{u}_1^{\mathrm{T}} + \hat{u}_4 \hat{u}_2^{\mathrm{T}} & \hat{u}_4 \hat{u}_4^{\mathrm{T}} - \tilde{u}_1 \tilde{u}_1^{\mathrm{T}} \end{bmatrix}, \end{aligned}$$

since $\hat{u}_3 = \tilde{u}_2$, due to the Hankel structure. Similar formulas are valid for the remaining part of ∇W, with corresponding definitions for $\hat{y}_1$, $\hat{y}_2$, etc., from $Y_{1,2s,N}$. Defining

$$g_1 = \begin{bmatrix} \hat{u}_2 \\ \hat{u}_4 \\ \hat{y}_2 \\ \hat{y}_4 \end{bmatrix}, \quad g_2 = \begin{bmatrix} 0 \\ \tilde{u}_1 \\ 0 \\ \tilde{y}_1 \end{bmatrix} = Z^{\mathrm{T}} H^{\mathrm{T}}(:, 1), \quad \hat{W} = \begin{bmatrix} \hat{u}_1 \hat{u}_1^{\mathrm{T}} & \hat{u}_1 \hat{u}_3^{\mathrm{T}} & \hat{u}_1 \hat{y}_1^{\mathrm{T}} & \hat{u}_1 \hat{y}_3^{\mathrm{T}} \\ \hat{u}_3 \hat{u}_1^{\mathrm{T}} & 0 & \hat{u}_3 \hat{y}_1^{\mathrm{T}} & 0 \\ \hat{y}_1 \hat{u}_1^{\mathrm{T}} & \hat{y}_1 \hat{u}_3^{\mathrm{T}} & \hat{y}_1 \hat{y}_1^{\mathrm{T}} & \hat{y}_1 \hat{y}_3^{\mathrm{T}} \\ \hat{y}_3 \hat{u}_1^{\mathrm{T}} & 0 & \hat{y}_3 \hat{y}_1^{\mathrm{T}} & 0 \end{bmatrix},$$

then

$$\nabla W = g_1 g_1^{\mathrm{T}} - g_2 g_2^{\mathrm{T}} + \hat{W},$$

where the symmetric matrix $\hat{W}$ has rank $2(m + \ell)$ at most. Indeed, using a symmetric block permutation, $\mathscr{P}$, the third block row and column of $\hat{W}$ can be interchanged with the second block row and column, respectively. Let $\mathscr{H}$ be an orthogonal matrix obtained as a product of $(m + \ell)$ Householder

transformations $\mathscr{H}_i$, of which the ith transformation annihilates the elements $(m+\ell+i+1{:}2(m+\ell)s, i)$ of $\mathscr{P}^{\mathrm{T}}\hat{W}\mathscr{P}$, and modifies its element $(m+\ell+i, i)$, then

$$\mathscr{H}^{\mathrm{T}}\mathscr{P}^{\mathrm{T}}\hat{W}\mathscr{P}\mathscr{H} = \begin{bmatrix} \hat{W}_1 & \hat{W}_2^{\mathrm{T}} & 0 \\ \hat{W}_2 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

where $\hat{W}_1 = \hat{W}_1^{\mathrm{T}}$ and $\hat{W}_2$ are $(m+\ell)\times(m+\ell)$, both possibly of maximal rank, and $\hat{W}_2$ is upper triangular. This is a similarity transformation, hence it preserves the eigenvalues. But the symmetric matrix

$$\begin{bmatrix} \hat{W}_1 & \hat{W}_2^{\mathrm{T}} \\ \hat{W}_2 & 0 \end{bmatrix}$$

has no zero eigenvalue when $\hat{W}_1$ and $\hat{W}_2$ have rank $(m+\ell)$ (since a zero eigenvalue will imply that $\hat{W}_1$ and/or $\hat{W}_2$ have rank strictly less than $(m+\ell)$). Therefore, this matrix, hence $\hat{W}$ too, can have rank $2(m+\ell)$, but not larger. Consequently, ∇W can be written as in the factored form (32), where $G \in \mathbb{R}^{2(m+\ell+1)\times 2(m+\ell)s}$. Two generators have already been obtained. The remaining generators are those of the symmetric matrix $\hat{W}$. □

A generator G is *proper* if its first column is zero except possibly for the first element. The fast QR factorization algorithm is based on the following result [11]. If

$$A = \begin{bmatrix} a_{11} & \mathbf{a}_{12}^{\mathrm{T}} \\ \mathbf{a}_{12} & A_{22} \end{bmatrix}, \quad a_{11} \in \mathbb{R},\ a_{11} > 0,\ \mathbf{a}_{12} \in \mathbb{R}^{n-1}$$

is a positive semi-definite matrix with proper generator $G = [G_1^{\mathrm{T}}\ G_2^{\mathrm{T}}]^{\mathrm{T}}$, where G_1 has one row, and the first column of G_2 is zero, then G_1 is the first row of the Cholesky factor of A, and the generator for $\hat{A} = A - G_1^{\mathrm{T}}G_1$ is given by $\hat{G}$, where

$$\hat{A} = \begin{bmatrix} 0 & \mathbf{0}^{\mathrm{T}} \\ \mathbf{0} & A_{22} - \mathbf{a}_{12}a_{11}^{-1}\mathbf{a}_{12}^{\mathrm{T}} \end{bmatrix}, \quad \hat{G} = \begin{bmatrix} G_1\hat{Z} \\ G_2 \end{bmatrix}$$

and $\hat{Z}$ has 1 on the first superdiagonal and 0 elsewhere. This result is used recursively in the *generalized Schur algorithm* to find all rows of the Cholesky factor of A. The reduction of a generator to a proper one can be done using Householder transformations and hyperbolic rotations. Details are given in [17] and in the references therein.

The technique summarized above directly finds the Cholesky factor R of the matrix $W = \tilde{W} + H^{\mathrm{T}}H$. For the MOESP approach, the same algorithm as for N4SID is used, but the first two block columns of the resulting upper triangular factor R are interchanged, and then retriangularized, exploiting the structure. This follows easily using (31).

3.2. Finding system order

Finding the system order cannot be generally done with an automatic procedure. This is possible in some instances, e.g., by detecting the largest logarithmic gap in the singular values of the matrix

R_o in (25). But, usually, the order could be decided after performing additional calculations with systems of various order, by checking out the prediction capabilities. The SLIDENT toolbox, and its associated demonstration package, offer facilities for such an approach. The QR factorization offer discussed in Section 3.1, and the computation of the SVD of the matrix R_o in (25) are performed only once.

3.2.1. Computation of the oblique projection

Finding the oblique projection (11) is an essential intermediate calculation in the N4SID algorithm (see (25), with $\mathcal{O}_s$ defined in Theorem 2).

Assume now that $H^{\mathrm{T}} = [L\ 0]\mathcal{Q}^{\mathrm{T}} = LQ^{\mathrm{T}}$ is an LQ factorization of H^{T}, i.e., L is lower triangular of order $2(m+\ell)s$ and $\mathcal{Q} = [Q\ Q_2]$ is orthogonal. Then, $[U_{1,2s,N}^{\mathrm{T}}\ Y_{1,2s,N}^{\mathrm{T}}] = H = QL^{\mathrm{T}} = QR$ is a QR factorization of H. Partition the factor R as

$$R = [R_{\mathrm{p}}^u\ R_{\mathrm{f}}^u\ R_{\mathrm{p}}^y\ R_{\mathrm{f}}^y], \quad \text{or also} \quad R = [R_{ij}], \quad i,j = 1{:}4, \tag{33}$$

where the subscripts p and f stand for "past" and "future" data, respectively, and the four block columns (and block rows, in the second partition) have ms, ms, ℓs, and ℓs columns (and rows), respectively, and define $R_{\mathrm{p}} = [R_{\mathrm{p}}^u\ R_{\mathrm{p}}^y]$. Similarly, partition L, hence $L_{\mathrm{p}}^u = (R_{\mathrm{p}}^u)^{\mathrm{T}}$, etc. According to the results in (i) and (ii), Lemma 1, with the definition (17) in Section 2, and with MOESP weightings in (20) and (21),

$$\begin{aligned}\mathcal{O}_s^M &= (Y_{\mathrm{f}}/U_{\mathrm{f}}^{\perp})(W_{\mathrm{p}}/U_{\mathrm{f}}^{\perp})^{\dagger}(W_{\mathrm{p}}/U_{\mathrm{f}}^{\perp})\\ &= [(L_{\mathrm{f}}^y/(L_{\mathrm{f}}^u)^{\perp})(L_{\mathrm{p}}/(L_{\mathrm{f}}^u)^{\perp})^{\dagger}(L_{\mathrm{p}}/(L_{\mathrm{f}}^u)^{\perp})\ 0]\mathcal{Q}.\end{aligned} \tag{34}$$

Note that if the SVD of F is $F = U\Sigma V^{\mathrm{T}}$, then for Q orthogonal, the SVD of FQ^{T} is $FQ^{\mathrm{T}} = U\Sigma(QV)^{\mathrm{T}}$. Since the subspace-based implemented algorithms only use the matrices U and Σ of the SVD of the oblique projection, the matrix $\mathcal{Q}$ in (34) can be disregarded. Also, the zero submatrix can be dropped, and all computations could be based on

$$\tilde{\mathcal{O}}_s^M = (L_{\mathrm{f}}^y/(L_{\mathrm{f}}^u)^{\perp})(L_{\mathrm{p}}/(L_{\mathrm{f}}^u)^{\perp})^{\dagger}(L_{\mathrm{p}}/(L_{\mathrm{f}}^u)^{\perp}) = r_2^{\mathrm{T}}(r_1^{\mathrm{T}})^{\dagger}r_1^{\mathrm{T}},$$

according to result (iii) in Lemma 1, where $r_1 = R_{\mathrm{p}} - R_{\mathrm{f}}^u X_1$ and $r_2 = R_{\mathrm{f}}^y - R_{\mathrm{f}}^u X_2$ are the residuals of the "minimum norm solutions" (in the sense in (iii)), X_1 and X_2, of the least-squares problems

$$\min\|R_{\mathrm{f}}^u X - R_{\mathrm{p}}\|_2 \quad \text{and} \quad \min\|R_{\mathrm{f}}^u X - R_{\mathrm{f}}^y\|_2, \tag{35}$$

respectively. Using the complete orthogonal factorization of r_1 [2],

$$r_1 = Q\begin{bmatrix} T_{11} & 0 \\ 0 & 0 \end{bmatrix} Z^{\mathrm{T}}P^{\mathrm{T}}, \quad T_{11} \in \mathbb{R}^{k\times k}, \text{ upper triangular, } \operatorname{rank}(T_{11}) = k = \operatorname{rank}(r_1),$$

where Q, Z, and P are orthogonal (P is a permutation matrix), $Q = [\overbrace{Q_1}^{k}\ Q_2]$, then

$$\tilde{\mathcal{O}}_s^M = r_2^{\mathrm{T}}Q\begin{bmatrix} T_{11}^{-\mathrm{T}} & 0 \\ 0 & 0 \end{bmatrix} Z^{\mathrm{T}}P^{\mathrm{T}}PZ\begin{bmatrix} T_{11}^{\mathrm{T}} & 0 \\ 0 & 0 \end{bmatrix} Q^{\mathrm{T}} = r_2^{\mathrm{T}}Q_1Q_1^{\mathrm{T}}.$$

Summarizing, $\tilde{\mathcal{O}}_s^M = r_2^{\mathrm{T}}Q_1Q_1^{\mathrm{T}}$, where Q_1 consists of the first $\operatorname{rank}(r_1)$ columns of Q. Neither of the two least-squares problems in (35) should actually be solved, but their residuals are needed.

These residuals could be obtained by simple computations with the orthogonal matrix from the QR factorization of R_f^u, applied to the right-hand side terms of the problems in (35), see [2]. The advantage is that only orthogonal transformations are used for finding $\tilde{\mathcal{O}}_s^M$, and therefore, the problem conditioning is preserved. Both least-squares problems have the same coefficient matrix, R_f^u, and it consists of two $ms \times ms$ submatrices, the second one being upper triangular; this structure is exploited.

3.3. Finding system matrices and Kalman gain

3.3.1. Computation of the matrices A and C

Let the SVD of R_o in (25) be written as in (18), hence $R_o = U_1 S_1 V_1^T$, with $U = [U_1\ U_2]$ orthogonal. Once the system order has been chosen, Γ_s is available from (19), and the matrices A and C can be obtained from (22).

3.3.2. Computation of the matrices B and D

From (14) written for Y_f, it follows that, asymptotically, $U_2^T Y_f / U_f U_f^\dagger = U_2^T Y_f U_f^\dagger = U_2^T H_s$, since $\{u_k\}$ is uncorrelated with the zero mean, stationary ergodic sequences $\{w_k\}$ and $\{v_k\}$, $U_2^T \Gamma_s = U_2^T U_1 S_1^{1/2} = 0$, and $U_f U_f^\dagger = I$ (using condition (2) in Theorem 2). The formula above is equivalent to $U_2^T H_s = J$, where $J := U_2^T L_f^y (L_f^u)^\dagger$, or, using the second partition of R in (33), $J = U_2^T R_{14}^T (R_{11}^T)^\dagger$, for MOESP.[6] With the definition of H_s in (16), this is a linear system in B and D, which can be written as

$$F \begin{bmatrix} D \\ B \end{bmatrix} := \begin{bmatrix} Q_1 & Q_2 & \cdots & Q_{s-1} & Q_s \\ Q_2 & Q_3 & \cdots & Q_s & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ Q_s & 0 & \cdots & 0 & 0 \end{bmatrix} \begin{bmatrix} I_\ell & 0 \\ 0 & \Gamma_{s-1} \end{bmatrix} \begin{bmatrix} D \\ B \end{bmatrix} = \begin{bmatrix} J_1 \\ J_2 \\ \vdots \\ J_s \end{bmatrix}, \tag{36}$$

where $Q_i = (U_2(\ell(i-1)+1{:}\ell i, :))^T$ is $(\ell s - n) \times \ell$, Γ_{s-1} is $(\ell s - \ell) \times n$, $J = [J_1\ J_2\ \cdots\ J_s]$, and J_i is $(\ell s - n) \times m$, for $i = 1{:}s$. Indeed, from (16),

$$H_s = \left[EX \begin{bmatrix} 0_{\ell \times (m-1)s} \\ H_{s-1} \end{bmatrix} \right], \quad E := \begin{bmatrix} I_\ell & 0 \\ 0 & \Gamma_{s-1} \end{bmatrix}, \quad X := \begin{bmatrix} D \\ B \end{bmatrix} \tag{37}$$

and H_{s-1} can be expressed similarly using H_{s-2}, and so on. Hence, $J = U_2^T H_s$ is

$$[J_1\ J_2\ \cdots\ J_s] = [Q_{1:s} EX\ [Q_{2:s}\ 0] EX\ \cdots\ [Q_s\ 0\ \cdots\ 0] EX], \tag{38}$$

which is equivalent to (36).

---

[6] Actually, the MOESP approach uses a more elaborate formula, proven in [30] for $N \to \infty$, $J := U_2^T R_{2c} R_{1c}^\dagger = U_2^T H_s$, where $R_{1c} = [R_{12}^T\ R_{22}^T\ R_{11}^T]$, and $R_{2c} = [R_{13}^T\ R_{23}^T\ R_{14}^T]$, with partition $R = (R_{ij})$. This formula gave better results in our experiments than using $U_2^T R_{14}^T (R_{11}^T)^\dagger$. In the SLICOT implementation, the matrix R_{1c}^T is triangularized using a specialized QR decomposition which exploits its structure.

Structure exploiting algorithm: By a block column permutation of the first matrix in F in (36), using a block matrix P which has zero blocks, except I_ℓ blocks on the anti-diagonal, the following equation is obtained:

$$\begin{bmatrix} Q_s & Q_{s-1} & \cdots & Q_2 & Q_1 \\ 0 & Q_s & \cdots & Q_3 & Q_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & Q_s \end{bmatrix} P \begin{bmatrix} I_\ell & 0 \\ 0 & \Gamma_{s-1} \end{bmatrix} \begin{bmatrix} D \\ B \end{bmatrix} = \begin{bmatrix} J_1 \\ J_2 \\ \vdots \\ J_s \end{bmatrix}, \tag{39}$$

for which a fast algorithm, based on a structure exploiting QR factorization, has been described in [21]. The product of the known matrices in the left-hand side is not computed, but the upper triangular block Toeplitz matrix is fastly triangularized. The transformations are applied to the right-hand side, and the matrices B and D are then found in two steps (not counting the block permutations). The results obtained using this fast algorithm are often sufficiently accurate, especially for the MOESP technique. However, in some applications the intermediate calculations for J are ill-conditioned, and the computed B and D could be inaccurate.

Kronecker product-based algorithm: To avoid computations with ill-conditioned L_{f}^u (or R_{1c}), the problem for finding B and D should be reformulated. It has been shown in [26] that all system matrices could be obtained by solving the following least-squares problem (with adapted notation):

$$\min_{A,C,\tilde{J}} \left\| \begin{bmatrix} \Gamma_{s-1}^\dagger Z_{s+2} \\ Y_{s+1,s+1,N+s} \end{bmatrix} - \left[\begin{bmatrix} A \\ C \end{bmatrix} \middle| \tilde{J} \right] \begin{bmatrix} \Gamma_s^\dagger Z_{s+1} \\ U_{\mathrm{f}} \end{bmatrix} \right\|_{\mathrm{F}}^2, \tag{40}$$

where $\|.\|_{\mathrm{F}}$ denotes the Frobenius norm,

$$Z_{s+1} = Y_{\mathrm{f}} \Big/ \begin{bmatrix} U_{\mathrm{p}} \\ U_{\mathrm{f}} \\ Y_{\mathrm{p}} \end{bmatrix} = L_{\mathrm{f}}^y(:, 1{:}l_1) Q_{\mathrm{f}}^y,$$

$$Z_{s+2} = Y_{s+2,2s,N+s+1} \Big/ \begin{bmatrix} U_{\mathrm{p}} \\ U_{\mathrm{f}} \\ Y_{1,s+1,N} \end{bmatrix} = L_{\mathrm{f}}^y(\ell+1{:}\ell s, 1{:}l_2) \bar{Q}_{\mathrm{f}}^y,$$

$l_1 := (2m+\ell)s$, $l_2 := (2m+\ell)s+\ell$, Q_{f}^y, $\bar{Q}_{\mathrm{f}}^y$ have orthonormal rows, and

$$\tilde{J} := \left[\begin{bmatrix} B \\ D \end{bmatrix} - \begin{bmatrix} A \\ C \end{bmatrix} \Gamma_s^\dagger \begin{bmatrix} D \\ \Gamma_{s-1} B \end{bmatrix} \middle| \begin{bmatrix} \Gamma_{s-1}^\dagger H_{s-1} \\ 0 \end{bmatrix} - \begin{bmatrix} A \\ C \end{bmatrix} \Gamma_s^\dagger \begin{bmatrix} 0 \\ H_{s-1} \end{bmatrix} \right], \tag{41}$$

which is linear in B and D. Note that (40) is equivalent to

$$\min_{A,C,\tilde{J}} \left\| \begin{bmatrix} \Gamma_{s-1}^\dagger L_{\mathrm{f}}^y(\ell+1{:}\ell s, 1{:}l_1) \\ L_{\mathrm{f}}^y(1{:}\ell, 1{:}l_1) \end{bmatrix} - \left[\begin{bmatrix} A \\ C \end{bmatrix} \middle| \tilde{J} \right] \begin{bmatrix} \Gamma_s^\dagger L_{\mathrm{f}}^y(:, 1{:}l_1) \\ L_{\mathrm{f}}^u(:, 1{:}l_1) \end{bmatrix} \right\|_{\mathrm{F}}^2. \tag{42}$$

Note also that, using the observability assumption and the notation in (37),

$$\begin{bmatrix} B \\ D \end{bmatrix} = \begin{bmatrix} 0 & I_n \\ I_\ell & 0 \end{bmatrix} \begin{bmatrix} D \\ B \end{bmatrix} = \begin{bmatrix} 0 & \Gamma_{s-1}^{\dagger} \\ I_\ell & 0 \end{bmatrix} \begin{bmatrix} I_\ell & 0 \\ 0 & \Gamma_{s-1} \end{bmatrix} \begin{bmatrix} D \\ B \end{bmatrix} = \begin{bmatrix} 0 & \Gamma_{s-1}^{\dagger} \\ I_\ell & 0 \end{bmatrix} EX.$$

Denoting $M = [M_1 \ \cdots \ M_{s-1}] := \Gamma_{s-1}^{\dagger}$, $\tilde{J} = [\tilde{J}_1 \ \cdots \ \tilde{J}_s]$, with $\tilde{J}_i = \tilde{J}_{:,(i-1)m+1:im} \in \mathbb{R}^{(n+\ell)\times m}$, $i = 1{:}s$, and defining

$$\begin{bmatrix} L_{1|1:s} \\ L_{2|1:s} \end{bmatrix} := \begin{bmatrix} A \\ C \end{bmatrix} \Gamma_s^{\dagger}, \quad \tilde{Q}_1 := \begin{bmatrix} -L_{1|1} \\ I_\ell - L_{2|1} \end{bmatrix}, \quad \tilde{Q}_i := \begin{bmatrix} M_{i-1} - L_{1|i} \\ -L_{2|i} \end{bmatrix}, \quad i = 2{:}s,$$

where $\tilde{Q}_i \in \mathbb{R}^{(n+\ell)\times\ell}$, $i = 1{:}s$, then $\tilde{J}$ in (41) can be rewritten as $\tilde{J} = [\tilde{Q}_1 \cdots \tilde{Q}_s]H_s$, and using the same procedure as for (38), we obtain

$$[\tilde{J}_1 \ \tilde{J}_2 \ \cdots \ \tilde{J}_s] = [\tilde{Q}_{1:s}EX \ [\tilde{Q}_{2:s} \ 0]EX \ \cdots \ [\tilde{Q}_s \ 0 \ \cdots \ 0]EX]. \tag{43}$$

Problem (42) is not solved as such. (SLIDENT N4SID technique solves part of it, computing $[A^{\mathrm{T}} \ C^{\mathrm{T}}]^{\mathrm{T}}$, but for the combined MOESP plus N4SID technique, A and C are already available.) Actually, denoting RHS the right-hand side term in (42), which is known, then $\tilde{J}L_{\mathrm{f}}^{u}(:, 1{:}l_1)$ is found from $\tilde{J}L_{\mathrm{f}}^{u}(:, 1{:}l_1) = \mathrm{RHS} - [A^{\mathrm{T}} \ C^{\mathrm{T}}]^{\mathrm{T}}\Gamma_s^{\dagger}L_{\mathrm{f}}^{y}(:, 1{:}l_1)$. From (43),

$$\begin{aligned}[\tilde{J}_1 \ \tilde{J}_2 \ \cdots \ \tilde{J}_s]L_{\mathrm{f}}^{u}(:, 1{:}l_0) = \tilde{Q}_{1:s}EXL_{\mathrm{f}}^{u}(1{:}m, 1{:}l_0) + [\tilde{Q}_{2:s} \ 0]EXL_{\mathrm{f}}^{u}(m+1{:}2m, 1{:}l_0) + \cdots \\ + [\tilde{Q}_s \ 0 \ \cdots \ 0]EXL_{\mathrm{f}}^{u}((m-1)s+1{:}ms, 1{:}l_0),\end{aligned} \tag{44}$$

where $l_0 := 2ms$ (the last ℓs zero columns have been suppressed). But $\mathscr{A}X\mathscr{B} = \mathscr{C}$ is equivalent to $(\mathscr{B}^{\mathrm{T}} \otimes \mathscr{A})\,\mathrm{vec}(X) = \mathrm{vec}(\mathscr{C})$, where $\otimes$ denotes the Kronecker product, and $\mathrm{vec}(\mathscr{M})$ is the vector obtained by stacking the columns of the matrix $\mathscr{M}$. From (44) it follows that B and D can be obtained by solving the problem

$$\tilde{T}\,\mathrm{vec}(X) = \mathrm{vec}(\tilde{J}L_{\mathrm{f}}^{u}(:, 1{:}l_0)), \quad \tilde{T} = \sum_{i=1}^{s} R_{\mathrm{f}}^{u}(1{:}l_0, (i-1)m+1{:}im) \otimes (\tilde{F}_i E),$$

where $X = [D^{\mathrm{T}} \ B^{\mathrm{T}}]^{\mathrm{T}}$, and $\tilde{F}_i$ is given by the ith block row of the matrix F in (36) with Q_j replaced by $\tilde{Q}_j$, $j = 1{:}s$. Note that $\tilde{T} \in \mathbb{R}^{2ms(n+\ell)\times m(n+\ell)}$. In SLIDENT implementation, this algorithm based on Kronecker products calculations solves a problem having half the size of the corresponding problem solved by the MATLAB N4SID code in [27]. Specifically, the number of rows of $\tilde{T}$ is halved by using the upper triangular factor of the QR factorization of R_{f}^{u} instead of R_{f}^{u}, and the corresponding transformed matrix $L_{\mathrm{f}}^{y} = (R_{\mathrm{f}}^{y})^{\mathrm{T}}$.

The N4SID algorithm incorporated in the SLIDENT toolbox differs from the published algorithm [25,27] in many aspects and details. However, if few changes are made in the original algorithm, they will essentially produce the same matrices A, B, C, D; there could be differences in the signs of elements, but the systems computed by N4SID and SLICOT codes are related by a similarity transformation. The changes are described in [22].

Simulation-based algorithm: An alternative algorithm is included for the computation of the matrices B and D. An extension and refinement of the method in [24,32] is used. Specifically, denoting

$$X = [(\mathrm{vec}(D^{\mathrm{T}}))^{\mathrm{T}}\ (\mathrm{vec}(B))^{\mathrm{T}}\ x_1^{\mathrm{T}}]^{\mathrm{T}},$$

where x_1 is the estimate of the initial state of (1), then it can be shown by lengthy calculations that X is the least-squares solution of the system $\mathscr{S}X = \mathrm{vec}(Y)$, with the matrix $\mathscr{S}$ defined by

$$\mathscr{S} = [\mathrm{diag}(U)\ \mathscr{Y}] := [\mathrm{diag}(U)\ y^{11}\ \cdots\ y^{n1}\ y^{12}\ \cdots\ y^{nm}\ P\Gamma],$$

where $\mathrm{diag}(U) := \mathrm{diag}(U,\ldots,U) \in \mathbb{R}^{\ell t \times \ell m}$ has ℓ block rows and columns. In this formula, $y^{ij} := [y^{ij}(1)^{\mathrm{T}}, y^{ij}(2)^{\mathrm{T}},\ldots, y^{ij}(t)^{\mathrm{T}}]^{\mathrm{T}}$, and these vectors are computed using the following model, for $j = 1{:}m$, and for $i = 1{:}n$,

$$\begin{aligned} &x^{ij}(k+1) = Ax^{ij}(k) + e_i u_j(k), \quad x^{ij}(1) = 0,\\ &y^{ij}(k) = Cx^{ij}(k), \quad k = 1{:}t, \end{aligned} \tag{45}$$

where e_i is the ith n-dimensional unit vector, Γ is given by

$$\Gamma = [C^{\mathrm{T}}\ (CA)^{\mathrm{T}}\ (CA^2)^{\mathrm{T}}\ \cdots\ (CA^{t-1})^{\mathrm{T}}]^{\mathrm{T}}$$

and P is a permutation matrix that groups together the rows of Γ depending on the same row c_j of C, namely

$$[c_j^{\mathrm{T}}\ (c_jA)^{\mathrm{T}}\ (c_jA^2)^{\mathrm{T}}\ \cdots\ (c_jA^{t-1})^{\mathrm{T}}]^{\mathrm{T}}$$

for $j = 1{:}\ell$. The first block column, $\mathrm{diag}(U)$, is not explicitly constructed, but its structure is exploited. The last block column is evaluated using powers of A with exponents 2^k, but no nonnecessary powers are computed (ℓt is not expanded to a power of 2). No permutations are explicitly applied. For efficiently computing the trajectories $\{y^{ij}(k)\}$ in (45), a similarity transformation reducing A to a real Schur form is used. A special QR decomposition of the matrix $\mathscr{S}$ is computed. Let $U = q[r^{\mathrm{T}}\ 0]^{\mathrm{T}}$ be the QR decomposition of U, if $m > 0$, where r is an $m \times m$ upper triangular matrix. Then, $\mathrm{diag}(q^{\mathrm{T}})$ is applied to $\mathscr{Y}$ and $\mathrm{vec}(Y)$. The block-rows of the transformed $\mathscr{S}$ and $\mathrm{vec}(Y)$ are implicitly permuted so that the transformed matrix $\mathscr{S}$ becomes

$$\begin{bmatrix} \mathrm{diag}(r) & \mathscr{Y}_1 \\ 0 & \mathscr{Y}_2 \end{bmatrix},$$

where $\mathscr{Y}_1$ has ℓm rows. Then, the QR decomposition of $\mathscr{Y}_2$ is computed (sequentially, if $m > 0$) and used to obtain B and x_1. The intermediate results and the QR decomposition of U are then needed to find D. If a triangular factor is too ill-conditioned, then its SVD is computed. SVD is not generally needed if the input sequence is sufficiently persistently exciting and t is large enough. If the matrix $\mathscr{Y}$ cannot be stored in the provided workspace, the QR decompositions of $\mathscr{Y}_2$ and U are computed sequentially. The calculations simplify if D and/or x_1 are not needed.

The simulation-based algorithm discussed above for estimating the matrices B and D is usually less efficient than the structure exploiting and Kronecker product-based algorithms, because a large

least-squares problem has to be solved. Its advantage is that the accuracy could be better, since the computed B and D are fitted to the input and output trajectories. However, if matrix A is unstable, the matrices B and D could be inaccurate.

When only x_1 should be determined, given the system matrices A, B, C, D, and the input–output sequences, a simpler method is implemented. This is an extension and refinement of the method in [32]. Specifically, the output $y_0(k)$ of the system for zero initial state is computed for $k = 1{:}t$ using the given model. Then, the following least-squares problem is solved for x_1

$$\Gamma x_1 = [(y(1) - y_0(1))^{\mathrm{T}}\ (y(2) - y_0(2))^{\mathrm{T}}\ \cdots\ (y(t) - y_0(t))^{\mathrm{T}}]^{\mathrm{T}}. \tag{46}$$

The coefficient matrix Γ is evaluated using powers of A with exponents 2^k. The QR decomposition of Γ is computed. If its triangular factor R is too ill-conditioned, then the singular value decomposition of R is used. If the matrix Γ cannot be stored in the provided workspace, the QR decomposition is computed sequentially. Various particular cases are dealt with in the implemented algorithms, in order to increase the efficiency.

Note that it is possible to use part of the input–output trajectories for estimating B, D, and/or x_1, and this could speed-up the calculations.

3.3.3. Computation of the covariances and Kalman gain

It should be noted that the residual of the least-squares problem in (42) can be used to find the covariance matrices (see (24)). Having the system and covariance matrices, and taking the assumptions of time-invariance and stationarity into account, the steady-state Kalman gain is found by solving the corresponding discrete-time algebraic Riccati equation for the optimal filtering problem corresponding to (1) and (2). These calculations are standard (see, e.g., [1]) and will not be further detailed. The efficiency of several Riccati solvers and their reliability (assuming that the system and covariance matrices are known) are discussed for instance in [20].

4. Numerical results

This section summarizes typical results obtained using the subspace-based techniques available in the SLIDENT toolbox. Except for Application 24, taken from [27], the data used is publicly available on the Database for Identification of Systems (DAISY) site `http://www.esat.kuleuven.ac.be/sista/daisy`, in order to increase the accessibility and reproducibility. This collection contains various data sets: process industry systems, mechanical systems, biomedical systems, environmental systems, thermic systems, simulators, and time series. All DAISY data sets are included in this investigation. Table 1 gives a summary description of the applications considered in this paper, indicating the number of inputs m, outputs ℓ, block rows s, data samples used t, and the selected order n. Actually, Application 19 has too few data (50 samples), but many outputs ($\ell = 13$); the available data have been replicated 11 times, getting 600 samples. The Applications 1 and 20 each include four distinct data sets. These sets have been concatenated for getting the results reported here, but the SLIDENT ability for sequential processing has also been successfully tried on these applications.

The numerical results have been obtained on an IBM PC computer at 500 MHz, with 128 MB memory, using Digital Visual Fortran V6.5 and nonoptimized BLAS. The latest version, MATLAB

Table 1
Summary description of applications

No.	Application	t	m	ℓ	s	n
1	Ethane–ethylene distillation column	4×90	5	3	5	4
2	Glass furnace	1247	3	6	10	5
3	120 MW power plant	200	5	3	10	8
4	Industrial evaporator	6305	3	3	10	4
5	Simulation data for a pH neutralization process	2001	2	1	15	6
6	Step response of a fractional distillation column	251	3	2	15	3
7	Industrial dryer	867	3	3	15	10
8	Liquid-saturated steam heat exchanger	4000	1	1	15	5
9	Test setup of an industrial winding process	2500	5	2	15	6
10	Continuous stirred tank reactor	7500	1	2	15	5
11	Model of steam generator	9600	4	4	15	9
12	Ball-and-beam	1000	1	1	20	2
13	Laboratory setup for a hair dryer	1000	1	1	15	4
14	CD-player arm	2048	2	2	15	8
15	Wing flutter	1024	1	1	20	6
16	Flexible robot arm	1024	1	1	20	4
17	Steel subframe flexible structure	8523	2	28	21	20
18	Cutaneous potential of a pregnant woman	2500	0	8	21	14
19	Tongue displacement shapes in pronunciation of English vowels	12×50	0	13	20	6
20	Western basin of Lake Erie	4×57	5	2	5	4
21	Heat flow through a two-layer wall	1680	2	1	20	3
22	Heating system	801	1	1	15	7
23	One hour Internet traffic at Berkeley Laboratory	99,999	0	1	8	2
24	Glass tubes	1401	2	2	20	8
25	Simulator of in vivo MRS signals	256	0	2	20	8

6.5.1.199709 (R13) has been employed. Similar, but less extensive results have been obtained on a Sun 4 sparc Ultra-2 computer, using operating system OS 5.6, and Sun WorkShop Compiler FORTRAN 77 5.0, and under MATLAB 5.3.0.10183 (R11) or MATLAB 6.1 (R12). See, for instance, [22,23].

The simplest calls have been used for standard calculations, e.g.,

```
[sys,K,rcnd]=slsolver(s,y,u,n,alg);
```

where `solver` is `moesp`, `n4sid`, `moen4`, or `moesm` (see Appendix). The notation `slsolver` with indices 1, 2, or 3, indicates the algorithm used in SLICOT implementation: fast Cholesky, fast QR, and standard QR, respectively. The alternative MATLAB code for comparison is the latest `n4sid` implementation [15], included in MATLAB 6.5.1. The notation `n4sid` with indices M or C denote the possible options for the weighting matrices, `'MOESP'` and `'CVA'`, respectively. [7]

[7] Our previous reports and preprints made comparisons with MOESP [10] (corresponding to `slmoesm`) and the robust N4SID code in [27]. The SLIDENT function `slmoesp` is a refined version of an older MATLAB MOESP code [29]. MOESP could not solve the identification problem for Application 17 (giving an "Out of memory" error message), and N4SID did not finish after 16 h of execution on the Sun machine.

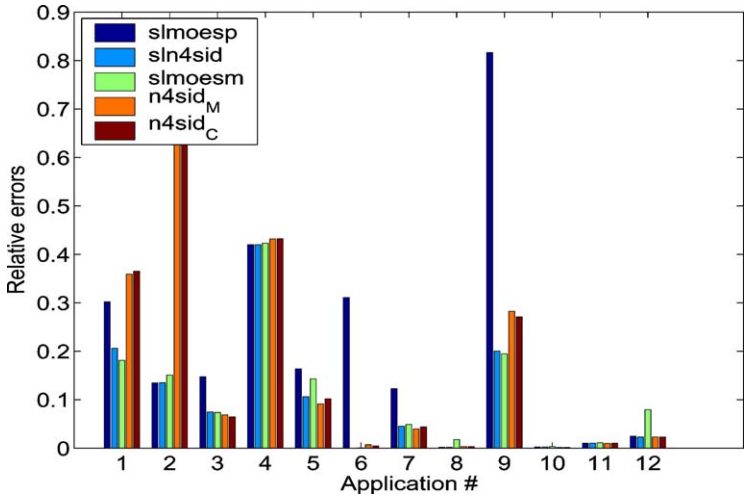

Fig. 1. Relative output errors, Applications 1–12. For each application, the bars from left to right display the results for `slmoesp`, `sln4sid`, `slmoesm`, $\texttt{n4sid}_M$, and $\texttt{n4sid}_C$, respectively.

Table 2
Euclidean norm of the relative output error vectors for applications solved by all solvers

Euclidean norm of relative output error				
`slmoesp`	`sln4sid` `slmoen4`	`slmoesm`	$\texttt{n4sid}_M$	$\texttt{n4sid}_C$
18.8420	1.2795	1.3048	1.6655	1.6312

Fig. 1 presents the relative output errors obtained using the SLIDENT and MATLAB `n4sid` codes for the first 12 applications and Table 2 gives the "cumulative" relative errors for applications solved by all solvers. For each solver, this error has been computed as the square root of the sum of squares of the relative output errors for all applications, i.e., it is the Euclidean norm of the vectors of relative errors. The reported relative output error has been computed with the following MATLAB formula: `err=norm(y-ye,1)/norm(y,1);`, where `ye` $:= \hat{y}$ denotes the estimated output of the system, evaluated using the estimated system matrices, A, B, C, and D, the given input trajectory, $\{u_k\}$, and an estimate of the initial state of the system, found, in turn, using A, B, C, D, and the given trajectories, $\{u_k\}$ and $\{y_k\}$. The SLIDENT results have been the same for all three implemented factorization algorithms. Both fast algorithms failed for Application 15, since the computed inter-correlation matrix W was not positive definite, but then the QR algorithm was automatically called. Moreover, `slmoen4` gave results identical with `sln4sid`, and, therefore, it is omitted from the bar graphs. The largest errors have been obtained for Applications 14 and 15 using `slmoesp`. These applications are difficult for standard MOESP approach, because ill-conditioned matrices appear during the computation of B and D, based on (39). However, there is no difficulty in solving these applications using the other implemented methods. Clearly, `sln4sid`, `slmoen4`, and `slmoesm` are equally good concerning the relative error. However, `sln4sid` could not solve the identification problem for Application 6; this application uses step functions as inputs, and

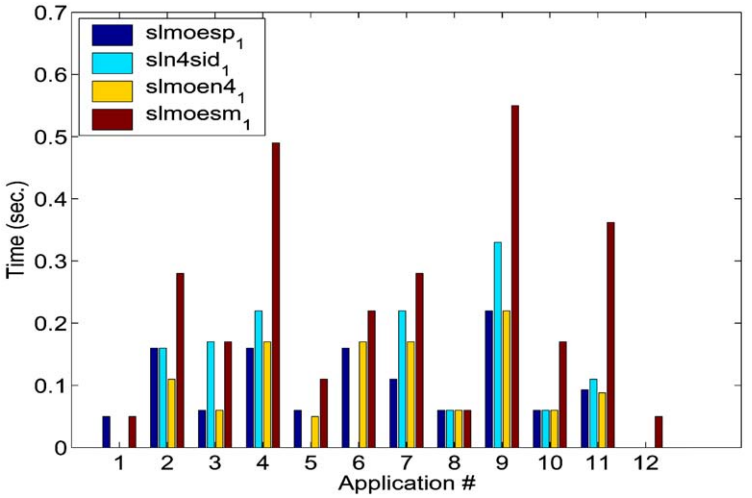

Fig. 2. Timings for fast Cholesky factorization algorithm, Applications 1–12. The times for Application 11 have been divided by 10. For each application, the bars from left to right display the results for `slmoesp`$_1$, `sln4sid`$_1$, `slmoen4`$_1$, and `slmoesm`$_1$, respectively.

consequently the condition 2 in Theorem 2 (persistently exciting u) is not fulfilled. This application was omitted when computing the results in Table 2.

The bar charts in Fig. 2 present the timing results when fast Cholesky factorization (algorithm 1) is used for the first 12 applications. The times for Application 11 have been divided by 10, in order to be comparable with those for other applications. The execution times for the fast QR factorization algorithm were similar, usually larger, except for Applications 17 and 18 (see Fig. 5).

The pie chart in Fig. 3 compares the cumulative timings (the sums of execution times for all applications solved by all solvers) for fast Cholesky algorithm and the `n4sid` code [15]; default options have been used for `n4sid`, except for the parameters `order`, `'N4Weight'`, and `'CovarianceMatrix'`, set to n in Table 1, `'MOESP'` (or `'CVA'`), and `'None'`, respectively. The value `'None'` for `'CovarianceMatrix'` ensures significantly (usually two or more times) smaller execution times than the default value, `'CovarianceMatrix'=[]`. Omitted applications are Application 2, which was not solved by SLICOT `sln4sid`, and Application 17, which could not be solved by `n4sid` with the mentioned options; `n4sid` gave an "Out of memory" error message, due to the too large size of the workspace needed for Application 17.

The Cholesky algorithm gave speed-up factors usually varying between 10 and 20, when comparing with SLICOT QR algorithm, and between 15 and 40, when comparing to the `n4sid` code. It is apparent from Fig. 2 that `slmoesp` and `slmoen4` were the most efficient methods, while `slmoesm` was less efficient for some applications. The general recommendation is to use `slmoen4` in conjunction with fast QR or Cholesky, since it is generally more accurate than `slmoesp`.

Fig. 4 presents the speed-up factors obtained for `slmoen4` with fast QR algorithm compared to `n4sid`, based on standard QR factorization; default options have been used for `n4sid`, except for parameters `order`, `'N4Weight'` and `'CovarianceMatrix'`, set to n, `'MOESP'`, and `'None'`, respectively. The use of the `'CVA'` for `'N4Weight'` gave similar results.

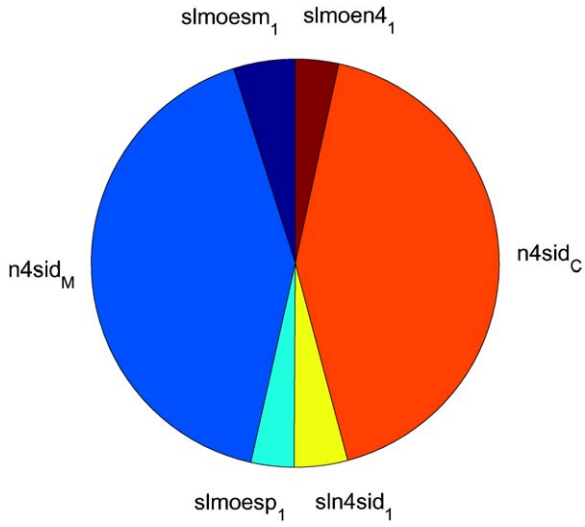

Fig. 3. Timing comparison: fast Cholesky versus MATLAB QR factorization algorithm.

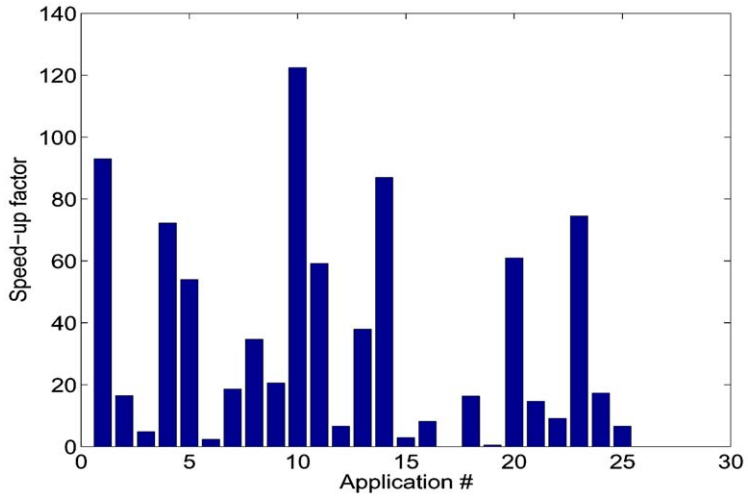

Fig. 4. `slmoen4` with fast QR versus MATLAB 6.5.1 `n4sid` with QR factorization (default options, except for `order` = n, `'N4Weight'` = `'MOESP'`, and `'CovarianceMatrix'` = `'None'`).

Fig. 5 shows the ratios between the CPU times for the fast QR algorithm and the Cholesky algorithm, which are close to one.

5. Conclusions

Algorithmic, implementation and numerical details concerning subspace-based techniques for linear multivariable system identification have been described. The techniques are implemented in the new system identification toolbox for the SLICOT Library—SLIDENT.

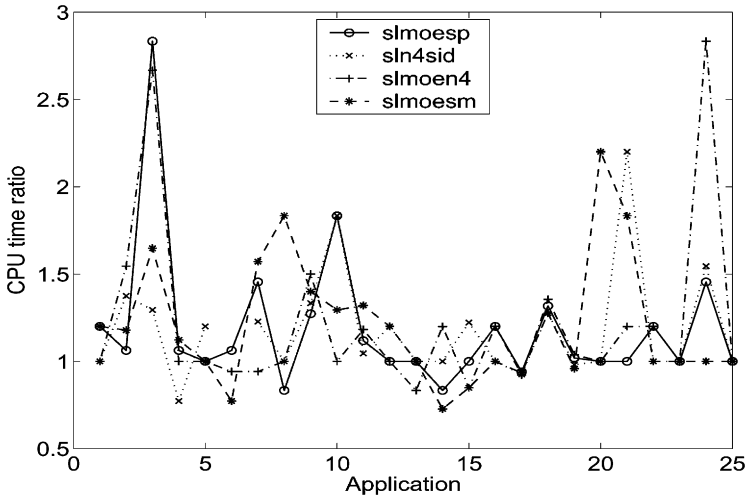

Fig. 5. CPU time ratios for fast QR algorithm over Cholesky algorithm.

This toolbox incorporates interfaces (MEX-files and M-files) to the MATLAB and Scilab environments, which improve the user-friendliness of the developed software. The results obtained show that the fast algorithmic variants included in the toolbox can frequently be used, and they are significantly more efficient than the standard QR factorization and the MATLAB codes. SLIDENT software can be used independently or in combination with available identification packages.

Future work includes further improving the performance and reliability of the SLICOT codes, and developing new algorithms or extensions for other problem classes, e.g., non-linear systems. Codes for Wiener-type systems, consisting in a linear part plus a static nonlinearity, have been already developed.

Acknowledgements

The authors are grateful to the anonymous referee for many valuable suggestions, which enable to improve the paper.

Appendix : User-friendly software components

The SLICOT linear system identification toolbox consists of 13 user-callable and computational routines, three MATLAB interfaces (MEX-files), and over 10 MATLAB M-files. The same interfaces could also be used in a Scilab environment [9]. The Fortran routines and the MATLAB MEX-files are described in [22]. The M-files call the MEX-files and they are included for user's convenience. On line information about the MEX-files and M-files can be obtained in the usual manner, e.g., by typing `help findR`, for getting details about the M-file `findR`. The MEX-files and M-files include several useful options, all having default values.

Table 3
SLIDENT method-oriented M-file interfaces

M-file	Function
`slmoesp`	finds system matrices and Kalman gain using MOESP technique
`sln4sid`	finds system matrices and Kalman gain using N4SID technique
`slmoen4`	finds system matrices and Kalman gain using combined MOESP and N4SID techniques: A and C via MOESP, B and D via N4SID
`slmoesm`	finds system matrices, Kalman gain, and initial state, using combined MOESP and system simulation techniques: A and C via MOESP, B and D via simulation

Most useful are four method-oriented M-files, briefly described in Table 3.
The calling sequences for these M-files are listed below:

```
[sys(,K,rcnd,R)]=slmoesp(s,Y(,U,n,alg,tol,printw)),

[sys(,K,rcnd,R)]=sln4sid(s,Y(,U,n,alg,tol,printw)),

[sys(,K,rcnd,R)]=slmoen4(s,Y(,U,n,alg,tol,printw)),

[sys(,K,rcnd,x0,R)]=slmoesm(s,Y(,U,n,alg,tol,printw));
```

where the parameters put inside the inner brackets are optional. Most of these parameters have clear meaning. If `n=0`, or `n=[]`, or `n` is omitted from the input parameters, the user is prompted to provide its value, after inspecting the singular values, shown as a bar plot. If $\mathtt{n} < 0$, then `n` is determined automatically. The parameter `alg` specifies the algorithm to compute the upper triangular factor R of the QR factorization of the matrix H: Cholesky algorithm on the correlation matrix (default), fast QR algorithm, or standard QR algorithm. The input parameter `tol` is a vector with two elements: `tol(1)` is the tolerance for estimating the rank of matrices, and `tol(2)` is the tolerance for estimating the system order. If $\mathtt{tol(1)} > 0$, the given value of `tol(1)` is used as a lower bound for the reciprocal condition numbers. If $\mathtt{tol(2)} \geqslant 0$, the estimate of `n` is indicated by the index of the last singular value greater than or equal to `tol(2)`. When $\mathtt{tol(2)} = 0$, then `s* eps * sval(1)` is used instead of `tol(2)`, where `sval(1)` is the maximal singular value, and `eps` is the relative machine precision. When $\mathtt{tol(2)} < 0$, the estimate is indicated by the index of the singular value that has the largest logarithmic gap to its successor. Default values are: `tol(1)=prod(size (`*matrix*`))* eps;tol(2)=-1`. The parameter `printw` should be set to 1 to print the warning messages, or to 0, otherwise (default). The output parameter `sys` is a discrete-time `ss` MATLAB system object, consisting in a state-space realization $\mathtt{sys} = (A, B, C, D)$. The parameter `rcnd` returns the reciprocal condition numbers, and, possibly, an error bound for the Riccati equation solution (needed for computing the Kalman gain K); these accuracy indicators are useful in assessing the reliability of the computed results. The output parameter `R` returns the processed factor R of the matrix H. It can be used for fast identification of systems of various orders, using, for instance, the

Table 4
Timing results (s) and relative error for Application 17 calling `slmoen4` in a loop; `s` = 31. The first `slmoen4` call took 199.98 s

n	Time	Relative error	n	Time	Relative error
16	6.76	0.37	24	10.05	0.20
20	8.29	0.35	28	12.03	0.05

following commands:

```
[sys,K,rcnd,R]=slmoen4(s,Y,U,n0,alg,tol,printw);
for n=n0+1 : min( n0+nf,s-1 )
  [sys,K,rcnd]=slmoen4(s,Y,U,n,R,tol,printw);
end
```

The data values for `Y` and `U` are not used inside the loop (only the size of `Y` is needed), but `R` replaces `alg`. Clearly, the systems of various orders (from `n0+1` to min(`n0+nf,s-1`)), should be used or saved (e.g., in a MATLAB cell array of systems) inside the loop. To illustrate the advantages of this feature, `slmoen4` was used for Application 17 in Section 4, with `s` = 31. This is a large problem, since $t = 8523$, $m = 2$, and $\ell = 28$. The first call took 199.98 s on the PC machine, but the loop calls were much faster, see Table 4. [8]

Shorter calls are possible, e.g.,

```
[sys,K]=slmoen4(s,Y);

sys=slmoen4(s,Y,U);
```

and similarly for the other M-files. The first call estimates the matrices A, C, and K of a stochastic system with no inputs; the order found should be confirmed by the user.

In addition, there are several M-files computing all or part of the system and covariance matrices and/or initial state. The calling sequences are listed below:

```
[R,n(,sval,rcnd)]=findR(s,Y(,U,meth,alg,jobd,tol,printw));

[sys(,K,Q,Ry,S,rcnd)]=findABCD(s,n,l,R(,meth,nsmpl,tol,printw));

[A,C(,rcnd)]=findAC(s,n,l,R(,meth,tol,printw));

[B(,D,K,Q,Ry,S,rcnd)]=findBDK(s,n,l,R,A,C(,meth,job,nsmpl,tol,printw));

[x0,B,D(,V,rcnd)]=findx0BD(A,C,Y(,U,withx0,withd,tol,printw));

[x0(,V,rcnd)]=inistate(sys,Y(,U,tol,printw));
```

Shorter calls are possible (see [22]).

[8] The `n4sid` code with default options, and MOESP weightings, could not solve the identification problem for $n = 28$, giving an "Out of memory" error after several hours of execution.

References

[1] B.D.O. Anderson, J.B. Moore, Optimal Filtering, Prentice-Hall, Englewood Cliffs, NJ, 1979.
[2] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, D. Sorensen, LAPACK Users' Guide, 3rd Edition, Series Software · Environments · Tools, SIAM, Philadelphia, 1999.
[3] P. Benner, V. Mehrmann, V. Sima, S. Van Huffel, A. Varga, SLICOT—A subroutine library in systems and control theory, in: B.N. Datta (Ed.), Applied and Computational Control, Signals, and Circuits, Vol. 1, Birkhäuser, Boston, 1999, pp. 499–539 (Chapter 10).
[4] Y.M. Cho, G. Xu, T. Kailath, Fast recursive identification of state space models via exploitation of displacement structure, Automatica 30 (1) (1994) 45–59.
[5] B. De Moor, P. Van Overschee, W. Favoreel, Numerical algorithms for subspace state-space system identification: an overview, in: B.N. Datta (Ed.), Applied and Computational Control, Signals, and Circuits, Vol. 1, Birkhäuser, Boston, 1999, pp. 247–311 (Chapter 6).
[6] J.J. Dongarra, J. Du Croz, I.S. Duff, S. Hammarling, Algorithm 679: A set of level 3 Basic Linear Algebra Subprograms, ACM Trans. Math. Software 16 (1990) 1–17, 18–28.
[7] J.J. Dongarra, J. Du Croz, S. Hammarling, R.J. Hanson, Algorithm 656: an extended set of Fortran Basic Linear Algebra Subprograms, ACM Trans. Math. Software 14 (1988) 1–17, 18–32.
[8] G.H. Golub, C.F. Van Loan, Matrix Computations, 3rd Edition, Johns Hopkins University Press, Baltimore, 1996.
[9] C. Gomez (Ed.), Engineering and Scientific Computing with SciLab, Birkhäuser, Boston, 1999.
[10] B. Haverkamp, M. Verhaegen, SMI Toolbox: State space model identification software for multivariable dynamical systems, Delft University of Technology, Report TUD/ET/SCE96.015, 1997.
[11] T. Kailath, A. Sayed, Displacement structure: theory and applications, SIAM Rev. 37 (1995) 297–386.
[12] W. Larimore, Canonical variate analysis in identification, filtering and adaptive control, in: Proceedings of the 29th Conference on Decision Control, CDC 90, Hawaii, 1990, pp. 596–604.
[13] C.L. Lawson, R.J. Hanson, D.R. Kincaid, F.T. Krogh, Basic Linear Algebra Subprograms for Fortran usage, ACM Trans. Math. Software 5 (1979) 308–323.
[14] L. Ljung, System Identification: Theory for the User, Prentice-Hall, Englewood Cliffs, NJ, 1987.
[15] L. Ljung, System Identification Toolbox for use with MATLAB, User's Guide, Version 5, The MathWorks, Inc, Natick, 2000.
[16] L. Ljung, T. McKelvey, Subspace identification from closed loop data, Signal Process. 52 (1996) 209–215.
[17] N. Mastronardi, D. Kressner, V. Sima, P. Van Dooren, S. Van Huffel, A fast algorithm for subspace state-space system identification via exploitation of the displacement structure, J. Comput. Appl. Math. 132 (2001) 71–81.
[18] M. Moonen, B. De Moor, L. Vandenberghe, J. Vandewalle, On- and off-line identification of linear state space models, Internat. J. Control 49 (1989) 219–232.
[19] K. Peternell, W. Scherrer, M. Deistler, Statistical analysis of novel subspace identification methods, Signal Process. 52 (1996) 161–177.
[20] V. Sima, Algorithms for Linear-Quadratic Optimization, Marcel Dekker, Inc., New York, 1996.
[21] V. Sima, Subspace-based algorithms for multivariable system identification, Stud. Informatics Control 5 (1996) 335–344.
[22] V. Sima, SLICOT linear systems identification toolbox, Katholieke Universiteit Leuven (ESAT/SISTA), SLICOT Working Note 2000-4, July 2000, Available `ftp://wgs.esat.kuleuven.ac.be/pub/WGS/REPORTS/`, file `SLWN2000-4.ps.Z`.
[23] V. Sima, S. Van Huffel, Performance investigation of SLICOT system identification toolbox, in: Proceedings of the European Control Conference, ECC 2001, Porto, 4–7 September 2001, pp. 3586–3591.
[24] V. Sima, A. Varga, RASP-IDENT: Subspace model identification programs, Institute for Robotics and System Dynamics, D-82230 Weßling, Technical Report TR R888-94, October 1994.
[25] P. Van Overschee, Subspace identification: theory–implementation–applications, Ph.D. Thesis, Department of Electrical Engineering, Katholieke Universiteit Leuven, 1995.
[26] P. Van Overschee, B. De Moor, N4SID: Two subspace algorithms for the identification of combined deterministic-stochastic systems, Automatica 30 (1994) 75–93.

[27] P. Van Overschee, B. De Moor, Subspace Identification for Linear Systems: Theory–Implementation–Applications, Kluwer Academic Publishers, Dordrecht, 1996.
[28] M. Verhaegen, Subspace model identification. Part 3: Analysis of the ordinary output-error state-space model identification algorithm, Internat. J. Control 58 (1993) 555–586.
[29] M. Verhaegen, State space model identification toolbox, Delft University of Technology, Technical Report, November 1993.
[30] M. Verhaegen, Identification of the deterministic part of MIMO state space models given in innovations form from input–output data, Automatica 30 (1994) 61–74.
[31] M. Verhaegen, P. Dewilde, Subspace model identification. Part 1: The output-error state-space model identification class of algorithms, Internat. J. Control 56 (1992) 1187–1210.
[32] M. Verhaegen, A. Varga, Some experience with the MOESP class of subspace model identification methods in identifying the BO105 helicopter, Institute for Robotics and System Dynamics, D-82230 Weßling, Technical Report TR R165-94, June 1994.
[33] M. Viberg, B. Wahlberg, B. Ottersten, Analysis of state space system identification methods based on instrumental variables and subspace fitting, Automatica 33 (9) (1997) 1603–1616.